

Distributed Decoding of Convolutional Network Error Correction Codes

Hengjie Yang and Wangmei Guo

Abstract

A Viterbi-like decoding algorithm is proposed in this paper for generalized convolutional network error correction coding. Different from classical Viterbi algorithm, our decoding algorithm is based on minimum error weight rather than the shortest Hamming distance between received and sent sequences. Network errors may disperse or neutralize due to network transmission and convolutional network coding. Therefore, classical decoding algorithm cannot be employed any more. Source decoding was proposed by multiplying the inverse of network transmission matrix, where the inverse is hard to compute. Starting from the *Maximum A Posteriori (MAP)* decoding criterion, we find that it is equivalent to the minimum error weight under our model. Inspired by Viterbi algorithm, we propose a Viterbi-like decoding algorithm based on minimum error weight of combined error vectors, which can be carried out directly at sink nodes and can correct any network errors within the capability of convolutional network error correction codes (CNECC). Under certain situations, the proposed algorithm can realize the distributed decoding of CNECC.

I. INTRODUCTION

Network coding is a new technique introduced in [1] which allows nodes to make the combination of multiple information before forwarding it. It is shown with large advantages in throughput, load equalization and security and so on, and has attracted lots of attention [2] [3]. Network error correction coding was first proposed by Cai & Yeung to correct errors caused by adversaries, which was then completely introduced in [4] [5]. They extended the Hamming bound, Singleton bound and Gilbert-Varshamov bound from classical error correction coding to network coding. Refined coding bounds for network error correction were given in [6]. Zhang studied network error correction in packet networks [7], where an algebraic definition of the minimum distance for linear network codes was introduced and the decoding problem was studied. Network error detection by random network coding has been studied by Ho *et al.* [8]. Jaggi *et al.* [9] have developed random algorithms for network error correction with various assumptions on adversaries. A new general framework for non-coherent network error correction was introduced in [10]. In their framework, messages are modulated as subspaces, so a code for non-coherent network error correction is also called a subspace code. Using rank-metric codes, nearly optimal subspace codes are constructed and decoding algorithms are also studied in [11].

This paper was submitted in part at the 2017 IEEE International Symposium on Information Theory, Aachen, Germany.

Hengjie Yang is with the School of Telecommunications Engineering, Xidian University, Xi'an, China (e-mail: yanghengjieyhj@gmail.com)

Wangmei Guo is with the State Key Lab of Integrated Services Networks, Xidian University, Xi'an, China (e-mail: wangmeiguo@xidian.edu.cn)

Convolutional network coding is shown to have advantages in field size, small decoding delay and so on, which is more suitable for practical communications [12] [13]. Convolutional network-error correcting coding was introduced in [14] in the context of coherent network coding for acyclic instantaneous or unit-delay networks. They presented a convolutional code construction for a given acyclic instantaneous or unit-delay memory-free network that corrects a given pattern of network errors. For the same network, if the network code changes, then the convolutional code obtained through their scheme may also change. They also consider the decoding. Decoding may be carried out at source or sink nodes based on the distance of equivalent received sequences. If the decoding needs to be done at the source node, the inverse of network transmission matrix is multiplied to the received sequences at a sink node, and then messages are decoded with the classical Viterbi algorithm. Distance measures for convolutional codes in rank metric were given in [15], and two constructions of (partial) unit memory ((P)UM) codes in rank metric based on the generator matrices of maximum rank distance codes were presented to correct network errors for non-coherent multi-shot networks. They also provided an efficient decoding algorithm based on rank-metric block decoders for error correction in random linear network coding.

In this paper, we consider the decoding for a generalized convolutional network error-correction code constructed by the extended method in [14], where network errors on each edge occur with the same probability and are separated by fixed timeslots. Source encoding is always needed due to the error correction purpose in networks.

However, given a delay-invariant, single source multicast network with a generalized multicast convolutional code, the ideal scenario is to decode CNECC in a distributed way while ensuring the decoded sequence is identical to the input sequence, which we refer to as the distributed decoding of CNECC. To realize it, we propose a Viterbi-like decoding algorithm based on minimum error weight and give a sufficient condition to show the feasibility of such decoding process.

The main contributions of this paper are as follows:

- A Viterbi-like decoding algorithm is proposed based on minimum error weight for generalized convolutional error correction coding.
- The algorithm can be carried out directly at sink nodes and no additional processing matrix is required.
- A sufficient condition of realizing the distributed decoding of CNECC is first given.

This paper is structured as follows. In Section II, definitions and notation for generalized convolutional network error correction coding as well as the relation between error and distance are given. Section III gives a sufficient condition of realizing the distributed decoding of CNECC. Section IV illustrates our Viterbi-like decoding algorithm and performance analysis. Some examples are shown in Section V and simulation results are given in Section VI. Finally, Section VII concludes this paper.

II. DEFINITIONS AND NOTATION

A. Network model

In this paper, we consider the finite directed network as in [7]. A finite directed graph G can be represented as $\{V, E\}$ where V is the set of all vertices in the network and E is the set of all edges in the network. A directed edge $e = (i, j)$ represents a channel leading from node i to node j where i is called the tail of e and

j is called the head of e , i.e., $\text{tail}(e) = i$ and $\text{head}(e) = j$. Channel e is called an outgoing channel for node i and an incoming channel for node j . For a node i , let $\text{In}(i) = \{e \in E : e \text{ is an incoming channel of } i\}$ and $\text{Out}(i) = \{e \in E : e \text{ is an outgoing channel of } i\}$.

Let S and T be two disjoint sets of V . The elements in S are called source nodes which only have outgoing channels. The elements in T are called sink nodes which only have incoming channels. The rest of nodes in set $I = V - S - T$ are called internal nodes. In this paper, we only consider single source networks and denote by s the unique source node.

We assume that each edge in the network has unit capacity (can carry utmost one symbol from \mathbb{F}_q) and capacity between nodes greater than one is modeled as parallel edges. A cut between node i and node j is a set of edges whose removal will disconnect i and j . For unit capacity channels, the capacity of a cut is regarded as the number of edges in it. For a sink node $t \in T$, let ω_t be the unicast capacity between s and t . Then, $\omega = \min_{t \in T} \omega_t$ is the max-flow min-cut capacity of the multicast connection. Here we let the information rate be ω symbols per time instant and we only consider multicast networks.

B. Network codes

We follow [3] in describing network codes. An ω -dimensional network code can be described by three matrices (over \mathbb{F}_q), $A_{\omega \times |E|}$, $K_{|E| \times |E|}(z)$, $B_{|E| \times \omega}^t$ (for every sink node $t \in T$). The details of them can be found in [3].

Definition 1 ([3]): The network transfer matrix, $M_t(z)$, corresponding to a sink node $t \in T$ for an ω -dimensional network code, is a full-rank (over the field of rationals $\mathbb{F}_q(z)$) $\omega \times \omega$ matrix defined as

$$M_t(z) := A(I_{|E| \times |E|} - K(z))^{-1} B^t = A F_t(z)$$

In this paper, we study network codes over \mathbb{F}_2 .

C. CNECC

We follow [14] in describing a CNECC. Assume the ω -dimensional network codes have been implemented in the given single source network. Note that the network may be a time-delay network, i.e. the network whose matrix K carries delay factor z . The following definitions describe a convolutional code for error correction in such networks.

Definition 2 ([14]): An input convolutional code, \mathcal{C}_s , is a convolutional code of rate k/ω ($k < \omega$) with an input generator matrix $G_I(z)$ implemented at the source of the network.

Definition 3: The output convolutional code, \mathcal{C}_t , corresponding to a sink node $t \in T$, is the k/ω ($k < \omega$) convolutional code generated by matrix $G_{O,t}(z)$ which is given as $G_{O,t}(z) = G_I(z)M_t(z)$, with $M_t(z)$ being the full-rank network transfer matrix corresponding to an ω -dimensional network code.

Definition 4: The free distance of the convolutional code \mathcal{C} is given as

$$d_{free}(\mathcal{C}) = \min\{w_H(\mathbf{v}(z)) | \mathbf{v}(z) = \mathbf{u}(z)G(z) \in \mathcal{C}, \mathbf{v}(z) \neq 0\}$$

where w_H indicates the Hamming weight over \mathbb{F}_q .

Definition 5 ([14]): Let \mathcal{C} be a rate b/c convolutional code with a generator matrix $G(z)$. Then corresponding to the information sequence $\mathbf{u}_0, \mathbf{u}_1, \dots (\mathbf{u}_i \in \mathbb{F}_q^b)$ and the codeword sequence $\mathbf{v}_0, \mathbf{v}_1, \dots (\mathbf{v}_i \in \mathbb{F}_q^c)$, we can associate

an encoder state sequence $\sigma_0, \sigma_1, \dots$, where σ_i indicates the content of the delay elements in the encoder at a time instant i . Define the set of j output symbols as $\mathbf{v}_{[0,j]} := [\mathbf{v}_0, \mathbf{v}_1, \dots, \mathbf{v}_{j-1}]$ and the set $S_{d_{free}}$ as follows.

$$S_{d_{free}} := \{\mathbf{v}_{[0,j]} | w_H(\mathbf{v}_{[0,j]}) < d_{free}(\mathcal{C}), \sigma_0 = \mathbf{0}, \forall j > 0\}$$

Clearly, the definition of $S_{d_{free}}$ excludes the possibility of a zero state in between, i.e., $\sigma_i \neq 0$ for any $0 < i \leq j$. We have that the set $S_{d_{free}}$ is invariant among the set of minimal convolutional encoders. We now define

$$T_{d_{free}}(\mathcal{C}) := \max_{\mathbf{v}_{[0,j]} \in S_{d_{free}}} j + 1$$

which thereby can be considered as a code property because of the fact that $S_{d_{free}}$ is invariant among minimal encoders.

Definition 6: An error pattern ρ is a subset of E which indicates the edges of the network in error. An error vector \mathbf{e} is a $1 \times |E|$ vector which indicates the error occurred at each edge. An error vector \mathbf{e} is said to match an error pattern if all nonzero components of \mathbf{e} occur only on the edges in ρ .

Let $\mathbf{x}(z)$, $\mathbf{y}(z)$, $\mathbf{e}(z)$ be the input sequence, output sequence and network error sequence, respectively. Thus, at any particular sink node $t \in T$, we have

$$\begin{aligned} \mathbf{y}(z) &= \mathbf{x}(z)G_I(z)M_t(z) + \mathbf{e}(z)F_t(z) \\ &= \mathbf{x}(z)G_{O,t}(z) + \mathbf{e}(z)F_t(z) \end{aligned} \quad (1)$$

In Section IV [14], the authors proposed a construction scheme that can correct a given set of error patterns as long as consecutive network errors are separated by a certain interval. The convolutional code constructed under their scheme is a CNECC. They also proposed two cases of decoding such CNECC under different conditions.

III. DISTRIBUTED DECODING OF CNECC

We now study the characteristics of matrix $F_t(z)$ at sink node $t \in T$. Assume the degree of F_t is $l_t = \deg(F_t(z))$. Thus, $F_t(z)$ can be written as $F_t(z) = \sum_{i=0}^{l_t} F_i z^i$ where F_{l_t} is a nonzero matrix. For a particular error vector \mathbf{e} , we can get its resulting combined error vector $(\mathbf{e}F_0 \ \mathbf{e}F_1 \ \dots \ \mathbf{e}F_{l_t})$ where $\mathbf{e}F_i$ ($0 \leq i \leq l_t$) is a $1 \times \omega$ subvector from (1). Combined error vector characterizes the impact of an error vector \mathbf{e} in the network. If all combined error vectors can be corrected, then all network errors can be consequently corrected. Let $\mathbf{E}'_i = (\mathbf{e}_i F_0 \ \mathbf{e}_i F_1 \ \dots \ \mathbf{e}_i F_{l_t})$ be the combined error vector generated by error vector \mathbf{e}_i in which only the i^{th} bit is one and the rest are all zeros. Let L be a collection of vectors in a linear space and $\langle L \rangle$ represent the subspace spanned by the vectors in L . Here we consider the subspace $\Delta(t, l_t) = \langle \{\mathbf{E}'_i : 1 \leq i \leq E\} \rangle$ at sink node $t \in T$.

Note that $\Delta(t, l)$ can also be regarded as a subspace determined by parameter l where $l \geq l_t$. For a given set of error patterns, assume $G_I(z)$ has been constructed using the scheme in [14]. We hope to find a minimum l such that $\Phi(t, l) \cap \Delta(t, l) = \{\mathbf{0}\}$ where $\mathbf{0}$ is a $1 \times \omega(l+1)$ zero vector and $\Phi(t, l)$ is the message subspace spanned by output convolutional codes generated by $\mathbf{x}_l(z)G_{O,t}(z)$ where $\mathbf{x}_l(z)$ are all possible input sequences from instant 0 to l , i.e., $(00 \dots 00)_l, (00 \dots 01)_l, \dots, (11 \dots 11)_l$.

Before discussing how to realize the distributed decoding of a CNECC, we give the following proposition.

Proposition 1: Assume l is available at sink node $t \in T$ such that $\Phi(t, l) \cap \Delta(t, l) = \{\underline{0}\}$. Then in any sliding window with length $l + 1$ on the output trellis of $G_{O,t}(z)$, at most one nonzero combined error vector exists as long as network errors are separated by $l + 1$ timeslots.

Proof: We use reduction to absurdity to prove this proposition. Before the proof, we assume all sliding window mentioned below share the same length $l + 1$. Apparently, proving all situations in first window hold true is enough as all situations in succeeding windows are equivalent to that in first window by removing the impact of the input sequence prior to current window. Assume in first window we find two different nonzero combined error vectors $e_1, e_2 \in \Delta(t, l)$ which implies that there exists two different input sequences x_1, x_2 satisfying $f(x_1) + e_1 = f(x_2) + e_2$ where $f(x) = x(z)G_{O,t}(z) \in \Phi(t, l)$. Given the closure property of $\Phi(t, l)$ and $\Delta(t, l)$, i.e., $f(x_1) - f(x_2) = f(x_1 - x_2) = f(x_3) \in \Phi(t, l)$ and $e_2 - e_1 = e_3 \in \Delta(t, l)$ where x_3, e_3 are another input sequences and nonzero combined error vector in first window respectively, it leads to $f(x_3) = e_3$ which contradicts to the condition $\Phi(t, l) \cap \Delta(t, l) = \{\underline{0}\}$. The proof is completed. ■

Proposition 1 indicates the uniqueness of nonzero combined error vector within each sliding window when network errors are separated by a certain interval. However, the crux of realizing the distributed decoding of CNECC is how to pinpoint the unique time instant when network errors occur. That is, the addition of two combined error vectors with one generating at current instant and another generating at a later instant may yield a certain correct output sequence. Under the circumstances, we cannot distinguish exactly when network errors occur as both time instants are possible. In fact, if an addition yields a correct output sequence with $v_0 = \mathbf{0}$, we can subjectively determine the network errors occur at the time instant of which is earlier than another. We call it error preposing which can enable the decoding algorithm still to proceed. The real issue is how to determine network errors when an addition yields a correct output sequence with $v_0 \neq \mathbf{0}$. The following proposition gives a sufficient condition that is able to realize the distributed decoding of CNECC.

Proposition 2: The distributed decoding of CNECC at sink node $t \in T$ can be realized when $G_{O,t}(z)$ is non-catastrophic and its free distance $d_{free} \geq 2\omega(l_t + 1) + 1$.

Proof: The main feature of non-catastrophic encoder is that only finite-length consecutive zero outputs exist. Similar to Definition 5, define $W(l)$ as follows.

$$W(l) := \min\{w_H(\mathbf{v}_{[0,l]}) | l \geq 0, \sigma_0 = \mathbf{0}\}$$

where $\mathbf{v}_{[0,l]} := \{\mathbf{v}_0, \mathbf{v}_1, \dots, \mathbf{v}_l\}$. Given the fact that $G_{O,t}(z)$ is non-catastrophic, $W(l)$ is an increasing function as l increases, though at some time instant $W(l)$ may stay the same temporarily. For $l \geq T_{d_{free}}$, $W(l) = d_{free}$. Given the fact that the addition of arbitrarily two combined error vectors (meaning that they can generate at different time instants) can affect at most $2\omega(l_t + 1)$ bits, there must exist a threshold l_{gate} such that $W(l_{gate}) \geq 2\omega(l_t + 1) + 1$. Let $l = l_{gate}$, therefore in the first sliding window $[0, l]$, none of the addition of arbitrarily two combined error vectors is identical to a certain correct output sequence, indicating that the exact time instant when network errors occur can be pinpointed. By using mathematical induction, network errors can be pinpointed in all subsequent windows which implies that the distributed decoding is realized. The proof is completed. ■

Proposition 2 gives a sufficient condition that can realize the distributed decoding of CNECC. In fact, introducing l_{gate} is to prove the feasibility of such decoding process. The real threshold value may be less than l_{gate} as the fundamental condition of realizing the distributed decoding of CNECC is that no addition of arbitrarily two combined error vectors can yield a certain correct output sequence with $v_0 \neq \mathbf{0}$ in the first sliding window. l_{gate} is merely a sufficient value that is able to meet this condition. An example is shown in Section V to clarify this point.

IV. DECODING ALGORITHM OF CNECC

Assume l has been determined satisfying $\Phi(t, l) \cap \Delta(t, l) = \{\underline{0}\}$ and we consider the following model.

- 1) all single edge errors have the same error probability p and network errors are subject to i.i.d. under this distribution.
- 2) all network errors are separated by $l + 1$ timeslots and only occur at prior $|x(z)|$ time instants.
- 3) $G_{O,t}(z), F_t(z)$ are available at each sink node.

Clearly, MAP is equivalent to the minimum error weight in this model so we propose the following decoding algorithm that is able to find the MAP path based on this model.

Algorithm 1 Decoding algorithm at sink node $t \in T$

```

1: Make reference table  $\Delta(t, l)$ . For each combined error vector, its weight is defined as that of its corresponding
   minimum error vector.
2: while new  $y_i$  ( $0 \leq i \leq |y(z)| - 1$ ) is received do
3:   if  $i \geq l$  then
4:     Calculate new combined error vectors  $(E_0, \dots, E_l)$  from previous vectors with finite weight.
5:     if multiple  $(E_0, \dots, E_l)$ s converge to the same output of the same next state then
6:       Select the one with minimum weight as father.
7:     end if
8:      $N \leftarrow$  the number of vectors in current window
9:     for  $j = 0$  to  $N - 1$  do
10:       $(E_0, \dots, E_l)$  is the  $j^{th}$  combined error vector.
11:      if  $E_0 = \mathbf{0}$  then
12:        Set its accumulative weight as its father's.
13:      else if  $(E_0, \dots, E_l) \notin \Delta(t, l)$  then
14:        Set its accumulative weight infinite.
15:      else
16:        Calculate its accumulative weight.
17:        Remove it on output trellis.
18:      end if
19:    end for
20:    if The input can be uniquely determined then
21:      Output the newly found input sequence.
22:    end if
23:    Slide current window one instant forward
24:  end if
25: end while

```

Remark 1: In the above pseudo-code, $|\mathbf{y}(z)| = |\mathbf{x}(z)| + \max\{\deg(G_{O,t}(z)), l\}$ since we need to encompass all combined error vectors completely. Compared to the decoding algorithm in [14], our algorithm has the following advantages. First, it can work directly at each sink node and no additional processing matrix has to be multiplied. Second, our algorithm can find the MAP path regardless of the error correction capability of $G_{O,t}(z)$ as in networks, MAP is equivalent to the minimum error path under our model. Third, the performance of the proposed algorithm is closely related to the characteristics of $F_t(z)$ and the free distance of $G_{O,t}(z)$. If conditions in Proposition 2 can be met at each sink node $t \in T$, the distributed decoding process can be realized on the whole network when all network errors are separated by $l_{\max} + 1$ timeslots where l_{\max} is the maximum of thresholds of all sink nodes.

In fact, the whole decoding process is quite similar to the classical Viterbi algorithm as we decode messages purely based on the metric of minimum error weight. Under our model, the impact of an error vector, i.e., its corresponding combined error vector, has no overlap with one another on output trellis so that the algorithm could determine its corresponding minimum error vector by searching the reference table.

Next, we study the time complexity of the proposed algorithm. Assume the reference table $\Delta(t, l)$ is available at each sink node $t \in T$ and at any particular time instant, the maximum processing time, including searching reference table, calculating Hamming weight, etc., is C . The total time complexity is $O(Cn)$.

V. ILLUSTRATIVE EXAMPLES

Example 1: To illustrate the concept introduced in above sections and to show the complete decoding process of the proposed algorithm, we check a simple directed cyclic network shown in Fig. 1. With the given network code,

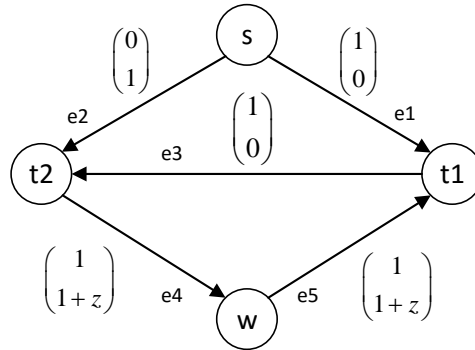


Fig. 1. Network G_1 , with $\omega = 2$

we thus have the network transfer matrices at sink t_1 and t_2 as follows

$$M_{t_1}(z) = \begin{bmatrix} 1 & 1 \\ 0 & 1+z \end{bmatrix} = AF_{t_1}(z)$$

where

$$F_{t_1}(z) = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 1+z & 1 & 1 & 1 \end{bmatrix}^T$$

TABLE I
THE SUBSPACE OF $\Phi(t_1, l)$ WITH $l = 2$

$\mathbf{x}(z)$ in window $[0, l]$	$\mathbf{x}(z)G_{O,t_1}(z)$ in window $[0, l]$
000	00 00 00
001	00 00 10
010	00 10 00
011	00 10 10
100	10 00 11
101	10 00 01
110	10 10 11
111	10 10 01

TABLE II
THE SUBSPACE OF $\Delta(t_1, l)$ WITH $l = 2$ (REFERENCE TABLE)

index	minimum error vector \mathbf{e}	$\mathbf{e}F_0$ $\mathbf{e}F_1$ $\mathbf{e}\mathbf{0}$	weight
0	(00000)	00 00 00	0
1	(10000)	11 00 00	1
2	(01000)	01 01 00	1
3	(00100) (00010) (00001)	01 00 00	1
4	(11000)	10 01 00	2
5	(10100) (10010) (10001)	10 00 00	2
6	(01100) (01010) (01001)	00 01 00	2
7	(11100) (11010) (11001)	11 01 00	3

and

$$M_{t_2}(z) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = AF_{t_2}(z)$$

where

$$F_{t_2}(z) = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{bmatrix}^T$$

As $M_{t_2}(z)$ is an identity matrix, therefore $G_{O,t_2}(z) = G_I(z)$ at sink node t_2 . Thus, we only consider sink node t_1 in later discussion. By using the construction scheme in [14], we choose $G_I(z) = [1 + z^2 \ 1 + z + z^2]$ with $d_{free} = 5$. Thus, $G_{O,t_1}(z) = G_I(z)M_{t_1}(z) = [1 + z^2 \ z^2 + z^3]$ with $d_{free} = 4$.

Next, we illustrate $l = 2$ satisfies the condition that $\Phi(t_1, l) \cap \Delta(t_1, l) = \{\underline{0}\}$. Let $F_{t_1}(z) = F_0 + F_1z + \mathbf{0}z^2$ where $\mathbf{0}$ is an $|E| \times \omega$ zero matrix. Thus, $\Phi(t_1, 1)$ and $\Delta(t_1, l)$ are given in Table I and II. Elements of $\Phi(t_1, l)$ with $l = 2$ are listed in column 2 of Table I and elements of $\Delta(t_1, l)$ with $l = 2$ are listed in column 3 of Table II. Thus, it can be easily checked that only $\underline{0}$ is in the intersection of $\Phi(t_1, l)$ and $\Delta(t_1, l)$.

Assume the input sequence $\mathbf{x}(z) = 1 + z^2 + z^5$ with $|\mathbf{x}(z)| = 6$. We set two types of $\mathbf{e}(z)$ where the former type contains indistinguishable network errors and the latter type is the opposite. They are given as follows.

$$\begin{aligned}\mathbf{e}_\alpha(z) &= (11000) + (00000)z + (00000)z^2 + (10100)z^3 + (00000)z^4 + (00000)z^5 \\ &= (1 + z^3 \ 1 \ z^3 \ 0 \ 0) \\ \mathbf{e}_\beta(z) &= (10000) + (00000)z + (00000)z^2 + (00100)z^3 + (00000)z^4 + (00000)z^5 \\ &= (1 \ 0 \ z^3 \ 0 \ 0)\end{aligned}$$

With equation (1), we have the following output sequences.

$$\begin{aligned}\mathbf{y}_\alpha(z) &= \mathbf{x}(z)G_{O,t_1}(z) + \mathbf{e}_\alpha(z)F_{t_1}(z) \\ &= (00) + (01)z + (01)z^2 + (11)z^3 + (11)z^4 + (11)z^5 + (00)z^6 + (11)z^7 + (01)z^8 \\ \mathbf{y}_\beta(z) &= \mathbf{x}(z)G_{O,t_1}(z) + \mathbf{e}_\beta(z)F_{t_1}(z) \\ &= (01) + (00)z + (01)z^2 + (00)z^3 + (11)z^4 + (11)z^5 + (00)z^6 + (11)z^7 + (01)z^8\end{aligned}$$

Now, we illustrate the decoding process when exerting network error sequence. At first, we list all combined error vectors (in decimal form) between $\mathbf{y}(z)$ and the output of trellis within the sliding window. The accumulative weight of each combined error vector at current sliding window is given in parentheses. Those with infinite weight are marked with a strikethrough which indicate they cease to extend. At the next time instant, new combined error vectors are derived from previous extendable states by removing the impact of its combined error vector and appending ω bits of new combined errors. At each time instant, all combined error vectors are listed in an increasing order of the internal state of the output trellis. Thus, the decoding process of $\mathbf{y}_\alpha(z)$ and $\mathbf{y}_\beta(z)$ are shown in Table III and IV, respectively.

In Table III, we can clearly see that two conflicts occur in window $[4, 6]$ and $[5, 7]$, respectively. In window $[4, 6]$, combined error vectors $(2 \ 1 \ 0)(5)$ and $(0 \ 0 \ 0)(4)$ both will converge to the same trellis output of the same state. So we select $(0 \ 0 \ 0)(4)$ due to its less weight. The case is same with $(0 \ 0 \ 0)(4)$ and $(3 \ 1 \ 0)(5)$ in window $[5, 7]$, we eventually select $(0 \ 0 \ 0)(4)$. At last, by backtracking the father path from $(0 \ 0 \ 0)(4)$ in window $[6, 8]$, we can obtain the MAP path $(1 \rightarrow 0 \rightarrow 1 \rightarrow 0 \rightarrow 0 \rightarrow 1)$ which is identical to $\mathbf{x}(z)$. In fact, $(1 \rightarrow 0 \rightarrow 1 \rightarrow 1 \rightarrow 0 \rightarrow 1)$ is also a possible path because its corresponding $\mathbf{e}(z) = (1 + z^5 \ 1 + z^5 \ z^5 \ 0 \ 0)$ that can yield the same $\mathbf{y}_\alpha(z)$. However its path weight is 5 more than that of the MAP path so that the algorithm discards it. We also can notice the whole decoding process of $\mathbf{y}_\alpha(z)$ must be carried out globally as the algorithm is unable to determine the unique MAP path in halfway.

In Table IV, we notice that the remaining combined error in window $[0, 2]$ and $[3, 5]$ are both unique suggesting that network errors are distinguishable in window so that the input sequence can be uniquely determined. Therefore, in window $[0, 2]$, the algorithm can promptly output (1) as the first newly found input sequence. Similarly, $(0 \rightarrow 1 \rightarrow 0)$ is output in window $[3, 5]$ as the second newly found sequence and $(0), (1), (0)$ are output in window $[4, 6], [5, 7], [6, 8]$, respectively. Eventually, the MAP path $(1 \rightarrow 0 \rightarrow 1 \rightarrow 0 \rightarrow 0 \rightarrow 1)$ is obtained in a distributed

TABLE III
DECODING PROCESS OF $y_\alpha(z)$ WITH $l = 2$

window	[0,2]	[1,3]	[2,4]	[3,5]	[4,6]	[5,7]	[6,8]
combined error vectors	0 1 1 (0)	1+1-3 (∞)	0 0 2 (3)	0 2 3 (3)	2-3-0 (∞)	1+1-3 (∞)	0 0 0 (4)
	2+1-2 (∞)	3 1 0 (3)	0 2 0 (2)	2-0-2 (∞)	0 1 1 (2)	3-3-2 (∞)	
	0 3 1 (0)	1-3-3 (∞)	0 0 0 (2)	0 0 1 (2)	0 3 3 (3)	0 0 0 (4)	
	2-3-2 (∞)	0 0 2 (2)	0 0 0 (3)	0 0 3 (3)	2+1-2 (∞)	3 1 0 (5)	
	0 1 3 (0)	3-3-0 (∞)	0 2 2 (2)	2-2-2 (∞)	2 1 0 (5)	1-3+1 (∞)	
	2 1 0 (2)	1+1+1 (∞)	0 0 2 (2)	0 2 1 (2)	0 0 0 (4)		
	0 3 3 (0)	3+1-2 (∞)		0 2 1 (3)	0 3 1 (2)		
	2-3-0 (∞)	1-3+1 (∞)		2 0 0 (4)	0 1 3 (3)		
		0 0 0 (2)		0 0 3 (2)	2-3-2 (∞)		
		3-3-2 (∞)		0 0 1 (3)			
				2-2-0 (∞)			
				0 2 3 (2)			

way. In fact, all combined errors within $e_\beta(z)$ are within the error capability of $G_{O,t_1}(z)$, which is why a distributed decoding can be realized.

TABLE IV
DECODING PROCESS OF $y_\beta(z)$ WITH $l = 2$

window	[0,2]	[1,3]	[2,4]	[3,5]	[4,6]	[5,7]	[6,8]
combined error vectors	1+0+1 (∞)	0 0 1 (1)	0 1 0 (1)	1+0-2 (∞)	0 0 0 (2)	0 0 0 (2)	0 0 0 (2)
	3-0-2 (∞)	0 0 3 (1)	0 3 0 (1)	3-0+1 (∞)			
	1-2+1 (∞)		0 1 2 (1)	1-2-2 (∞)			
	3-2-2 (∞)		0 3 2 (1)	3-2+1 (∞)			
	1+0-3 (∞)			1 0 0 (2)			
	3 0 0 (1)			3-0-3 (∞)			
	1-2-3 (∞)			1-2-0 (∞)			
	3-2-0 (∞)			3-2-3 (∞)			

Example 2: Assume $G_I(z) = (1 + z^2 + z^3 \ 1 + z + z^2 + z^3)$ and the network is the same as in Example 1. Thus, the output generators at sink node t_1, t_2 are as follows.

$$G_{O,t_1}(z) = G_I(z)M_{t_1}(z) = [1 + z^2 + z^3 \ z^2 + z^3 + z^4]$$

$$G_{O,t_2}(z) = G_I(z)M_{t_2}(z) = [1 + z^2 + z^3 \ 1 + z + z^2 + z^3]$$

Both output generators share the same free distance of 6. All elements with $v_0 \neq \mathbf{0}$ in window $[0, 6]$ of $G_{O,t_1}(z)$ and in window $[0, 3]$ of $G_{O,t_2}(z)$ are given in Table V and Table VI. For sink node t_1 , it can be checked that no addition of arbitrarily two combined error vectors in $\Delta(t_1, l)$ is identical to a certain element in Table V when $l = 6$. For sink node t_2 , the case is same with combined error vectors in $\Delta(t_2, l)$ and elements in Table VI when $l = 3$. It

TABLE V
ELEMENTS WITH $\mathbf{v}_0 \neq \mathbf{0}$ IN $\Phi(t_1, l)$ WITH $l = 6$

$\mathbf{x}(z)G_{O,t_1}(z)$ in window $[0, l]$	$\mathbf{x}(z)G_{O,t_1}(z)$ in window $[0, l]$	$\mathbf{x}(z)G_{O,t_1}(z)$ in window $[0, l]$	$\mathbf{x}(z)G_{O,t_1}(z)$ in window $[0, l]$
10 00 11 11 01 00 00	10 00 01 11 10 11 01	10 10 11 00 10 01 00	10 10 01 00 01 10 01
10 00 11 11 01 00 10	10 00 01 11 10 11 11	10 10 11 00 10 01 10	10 10 01 00 01 10 11
10 00 11 11 01 10 00	10 00 01 11 10 01 01	10 10 11 00 10 11 00	10 10 01 00 01 00 01
10 00 11 11 01 10 10	10 00 01 11 10 01 11	10 10 11 00 10 11 10	10 10 01 00 01 00 11
10 00 11 11 11 00 11	10 00 01 11 00 11 10	10 10 11 00 00 01 11	10 10 01 00 11 10 10
10 00 11 11 11 00 01	10 00 01 11 00 11 00	10 10 11 00 00 01 01	10 10 01 00 11 10 00
10 00 11 11 11 10 11	10 00 01 11 00 01 10	10 10 11 00 00 11 11	10 10 01 00 11 00 10
10 00 11 11 11 10 01	10 00 01 11 00 01 00	10 10 11 00 00 11 01	10 10 01 00 11 00 00
10 00 11 01 01 11 11	10 00 01 01 10 00 10	10 10 11 10 10 10 11	10 10 01 10 01 01 10
10 00 11 01 01 11 01	10 00 01 01 10 00 00	10 10 11 10 10 10 01	10 10 01 10 01 01 00
10 00 11 01 01 01 11	10 00 01 01 10 10 10	10 10 11 10 10 00 11	10 10 01 10 01 11 10
10 00 11 01 01 01 01	10 00 01 01 10 10 00	10 10 11 10 10 00 01	10 10 01 10 01 11 00
10 00 11 01 11 11 00	10 00 01 01 00 00 01	10 10 11 10 00 10 00	10 10 01 10 11 01 01
10 00 11 01 11 11 10	10 00 01 01 00 00 11	10 10 11 10 00 10 10	10 10 01 10 11 01 11
10 00 11 01 11 01 00	10 00 01 01 00 10 01	10 10 11 10 00 00 00	10 10 01 10 11 11 01
10 00 11 01 11 01 10	10 00 01 01 00 10 11	10 10 11 10 00 00 10	10 10 01 10 11 11 11

TABLE VI
ELEMENTS WITH $\mathbf{v}_0 \neq \mathbf{0}$ IN $\Phi(t_2, l)$ WITH $l = 3$

$\mathbf{x}(z)G_{O,t_2}(z)$ in window $[0, l]$	$\mathbf{x}(z)G_{O,t_2}(z)$ in window $[0, l]$
11 01 11 11	11 10 10 00
11 01 11 00	11 10 10 11
11 01 00 10	11 10 01 01
11 01 00 01	11 10 01 10

implies that if all network errors are separated by 7 timeslots, our decoding algorithm is able to decode messages in a distributed way at both t_1 and t_2 . We can notice that although the free distance of two output generators does not meet the condition in Proposition 2, the distributed decoding can still be realized.

VI. SIMULATION RESULTS OF THE DECODING ALGORITHM

A. A probabilistic error model

We define a probabilistic error model for a single source network $G(V, E)$ by defining the probabilities of any set of i ($1 \leq i \leq |E|$) edges of the network being in error at any given time instant as follows. Across time instants, assume that the network errors are subject to i.i.d. according to this distribution.

$$Pr(i \text{ network edges being in error}) = p^i \quad (2)$$

$$Pr(\text{no edges are in error}) = q \quad (3)$$

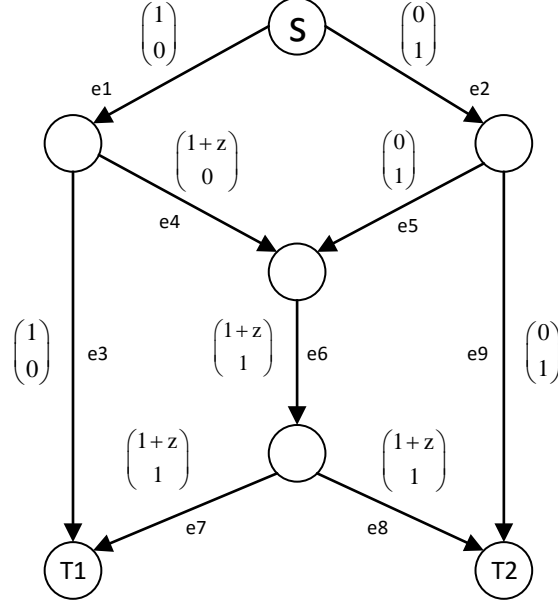


Fig. 2. Butterfly network G_2 , with $\omega = 2$

where $1 \leq i \leq |E|$ and $0 \leq p, q \leq 1$ are real numbers indicating the probability of any single network error in the network and probability of no network error, respectively, such that $q + \sum_{i=1}^{|E|} p^i = 1$.

B. Simulations on the butterfly network

We simulate the proposed algorithm in the classical butterfly network shown in Fig. 2. The transfer matrices for sink node t_1, t_2 are given as follows.

$$M_{t_1}(z) = \begin{bmatrix} 1 & 1+z \\ 0 & 1 \end{bmatrix} = AF_{t_1}(z)$$

where

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$F_{t_1}(z) = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1+z & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \end{bmatrix}^T$$

and

$$M_{t_2}(z) = \begin{bmatrix} 1+z & 0 \\ 1 & 1 \end{bmatrix} = AF_{t_2}(z)$$

where

$$F_{t_2}(z) = \begin{bmatrix} 1+z & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}^T$$

Let $l_t = \deg(F_t(z))$. The subspaces of $\Delta(t, l_t)$ for sink node t_1 and t_2 are shown as in Table VII and VIII, respectively. Thus, all $\Delta(t, l)$ with $l \geq l_t$ can be derived from $\Delta(t, l_t)$ by appending $\mathbf{0}$ to each combined error vector in the table.

TABLE VII
THE SUBSPACE OF $\Delta(t_1, l_{t_1})$ WITH $l_{t_1}=1$

index	minimum error vector e	eF_0 eF_1	weight
0	(000000000)	00 00	0
1	(100000000)	11 01	1
2	(001000000)	10 00	1
3	(010000000) (000100000) (000010000) (000001000) (000000100)	01 00	1
4	(101000000)	01 01	2
5	(110000000) (100100000) (100010000) (100001000) (100000100)	10 01	2
6	(011000000) (001100000) (001010000) (001001000) (001000100)	11 00	2
7	(111000000) (101100000) (101010000) (101001000) (101000100)	00 01	3

TABLE VIII
THE SUBSPACE OF $\Delta(t_2, l_{t_2})$ WITH $l_{t_2} = 1$

index	minimum error vector e	eF_0 eF_1	weight
0	(000000000)	00 00	0
1	(100000000)	10 10	1
2	(010000000)	11 00	1
3	(000000001)	01 00	1
4	(000100000) (000010000) (000001000) (000000010)	10 00	1
5	(110000000)	01 10	2
6	(100000001)	11 10	2
7	(100100000) (100010000) (100001000) (100000010)	00 10	2

Let $G_I(z) = [1 + z^2 + z^3 + z^4 \ 1 + z + z^4]$. The corresponding output generator matrices at sink node t_1 and t_2 are given as follows.

$$G_{O,t_1}(z) = G_I(z)M_{t_1}(z) = [1 + z^2 + z^3 + z^4 \ z^2 + z^4 + z^5]$$

$$G_{O,t_2}(z) = G_I(z)M_{t_2}(z) = [z^2 + z^4 + z^5 \ 1 + z + z^4]$$

It can be easily checked that $l = 2$ satisfying the basic condition that $\Phi(t, l) \cap \Delta(t, l) = \{\mathbf{0}\}$ for both sink nodes. Let the input sequences be randomly given and error vectors be generated continuously in time under the probabilistic

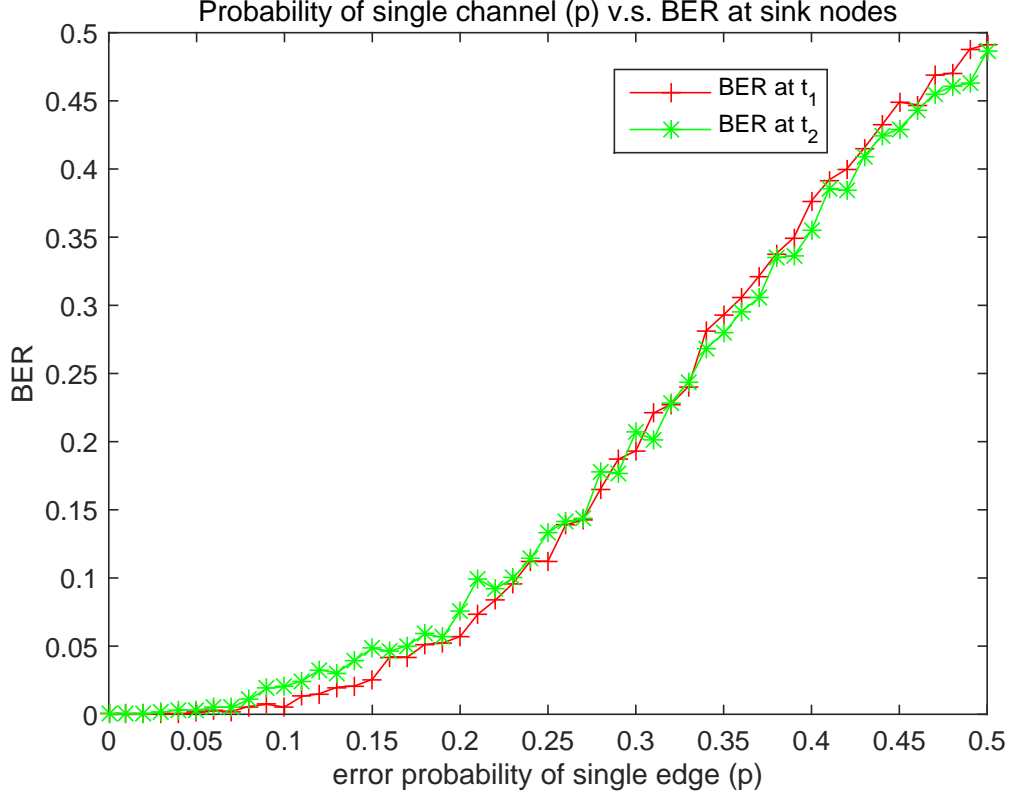


Fig. 3. BER at both sink nodes

error model (but the decoding algorithm assumes they have been separated by a certain interval l), we compare the decoded sequence obtained by the proposed decoding algorithm with the original input sequence and then compute the average BER given the single edge error probability p . The simulation result is shown in Fig. 3.

In Fig. 3, it can be easily observed that the BER performances of our decoding algorithm at sink node t_1 and t_2 are likely. They both have the following features. In the region where $0 \leq p \leq 0.16$, due to the small single edge error probability, network errors are separated by a sufficient timeslots on the output trellis with a high probability. Therefore the condition of distributed decoding of CNECC is easily satisfied resulting in a better BER performance. In the region where $0.16 < p \leq 0.5$, the network errors on output trellis get closer as the single edge error probability increases. Therefore the condition of distributed decoding of CNECC is no longer satisfied which results in a worse BER performance. However, compared to the simulation results in [14], we obtain relatively the same BER performance without increasing the decoding complexity by directly decoding messages at sink nodes under the criterion of minimum error weight.

VII. CONCLUSION

In this paper, a Viterbi-like decoding algorithm based on minimum error weight for generalized convolutional coding is proposed and a sufficient condition of realizing distributed decoding of CNECC is first given. The distributed decoding of CNECC is the ideal scenario we strive to pursue. However, the following issues regarding

this process still remain open and will be studied in future work. First, how to design a network encoding matrix that is capable of realizing the distributed decoding with a smaller window length. Second, how to select a suitable input generator matrix such that all output generator matrices of all sink nodes are non-catastrophic. Third, how to find a sufficient condition less restricted by d_{free} to realize the distributed decoding of CNECC.

REFERENCES

- [1] R. Ahlswede, N. Cai, S. Y. R. Li, and R. W. Yeung, "Network information flow," *IEEE Transactions on Information Theory*, vol. 46, no. 4, pp. 1204–1216, Jul 2000.
- [2] S. Y. R. Li, R. W. Yeung, and N. Cai, "Linear network coding," *IEEE Transactions on Information Theory*, vol. 49, no. 2, pp. 371–381, Feb 2003.
- [3] R. Koetter and M. Medard, "An algebraic approach to network coding," *IEEE/ACM Transactions on Networking*, vol. 11, no. 5, pp. 782–795, Oct 2003.
- [4] R. W. Yeung and N. Cai, "Network error correction, part i: Basic concepts and upper bounds," *Communications in Information and Systems*, vol. 6, no. 1, pp. 19–36, 2006.
- [5] —, "Network error correction, part ii: Lower bounds," *Communications in Information and Systems*, vol. 6, no. 1, pp. 37–54, 2006.
- [6] S. Yang and R. W. Yeung, "Refined coding bounds for network error correction," in *Information Theory for Wireless Networks, 2007 IEEE Information Theory Workshop on*, July 2007, pp. 1–5.
- [7] Z. Zhang, "Linear network error correction codes in packet networks," *IEEE Transactions on Information Theory*, vol. 54, no. 1, pp. 209–218, Jan 2008.
- [8] T. Ho, B. Leong, R. Koetter, M. Medard, M. Effros, and D. R. Karger, "Byzantine modification detection in multicast networks using randomized network coding," in *Information Theory, 2004. ISIT 2004. Proceedings. International Symposium on*, June 2004, pp. 144–.
- [9] S. Jaggi, M. Langberg, S. Katti, T. Ho, D. Katabi, M. Medard, and M. Effros, "Resilient network coding in the presence of byzantine adversaries," *IEEE Transactions on Information Theory*, vol. 54, no. 6, pp. 2596–2603, June 2008.
- [10] R. Koetter and F. R. Kschischang, "Coding for errors and erasures in random network coding," *IEEE Transactions on Information Theory*, vol. 54, no. 8, pp. 3579–3591, Aug 2008.
- [11] D. Silva, F. R. Kschischang, and R. Koetter, "A rank-metric approach to error control in random network coding," *IEEE Transactions on Information Theory*, vol. 54, no. 9, pp. 3951–3967, Sept 2008.
- [12] S. Y. R. Li, Q. T. Sun, and Z. Shao, "Linear network coding: Theory and algorithms," *Proceedings of the IEEE*, vol. 99, no. 3, pp. 372–387, March 2011.
- [13] W. Guo, X. Shi, N. Cai, and M. Medard, "Localized dimension growth: A convolutional random network coding approach to managing memory and decoding delay," *IEEE Transactions on Communications*, vol. 61, no. 9, pp. 3894–3905, September 2013.
- [14] K. Prasad and B. S. Rajan, "Network error correction for unit-delay, memory-free networks using convolutional codes," in *2010 IEEE International Conference on Communications*, May 2010, pp. 1–6.
- [15] A. Wachter-Zeh, M. Stinner, and V. Sidorenko, "Convolutional codes in rank metric with application to random network coding," *IEEE Transactions on Information Theory*, vol. 61, no. 6, pp. 3199–3213, June 2015.